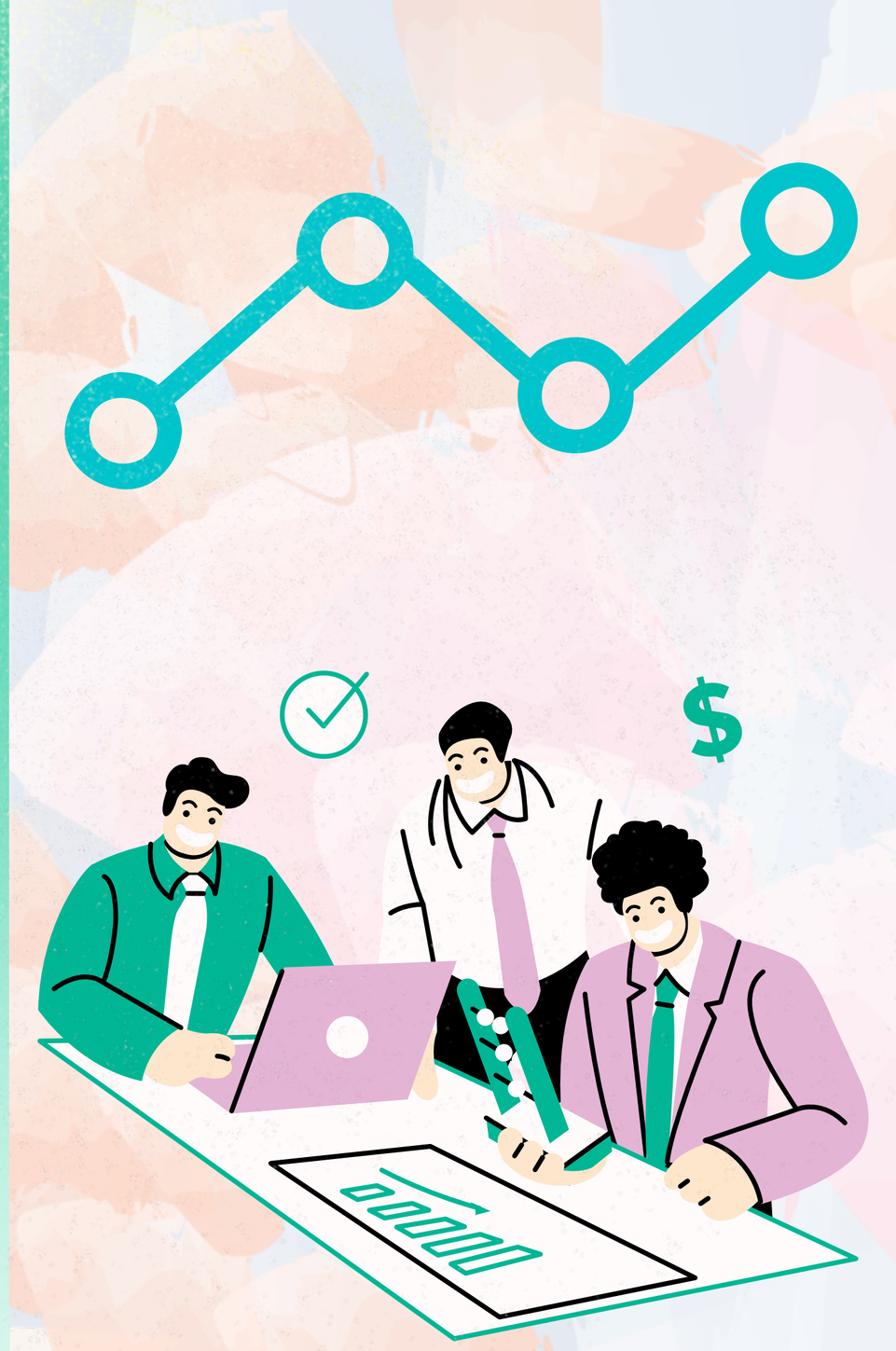
Data Science

Project Presentation by Team **Votrix**



Problem Statement

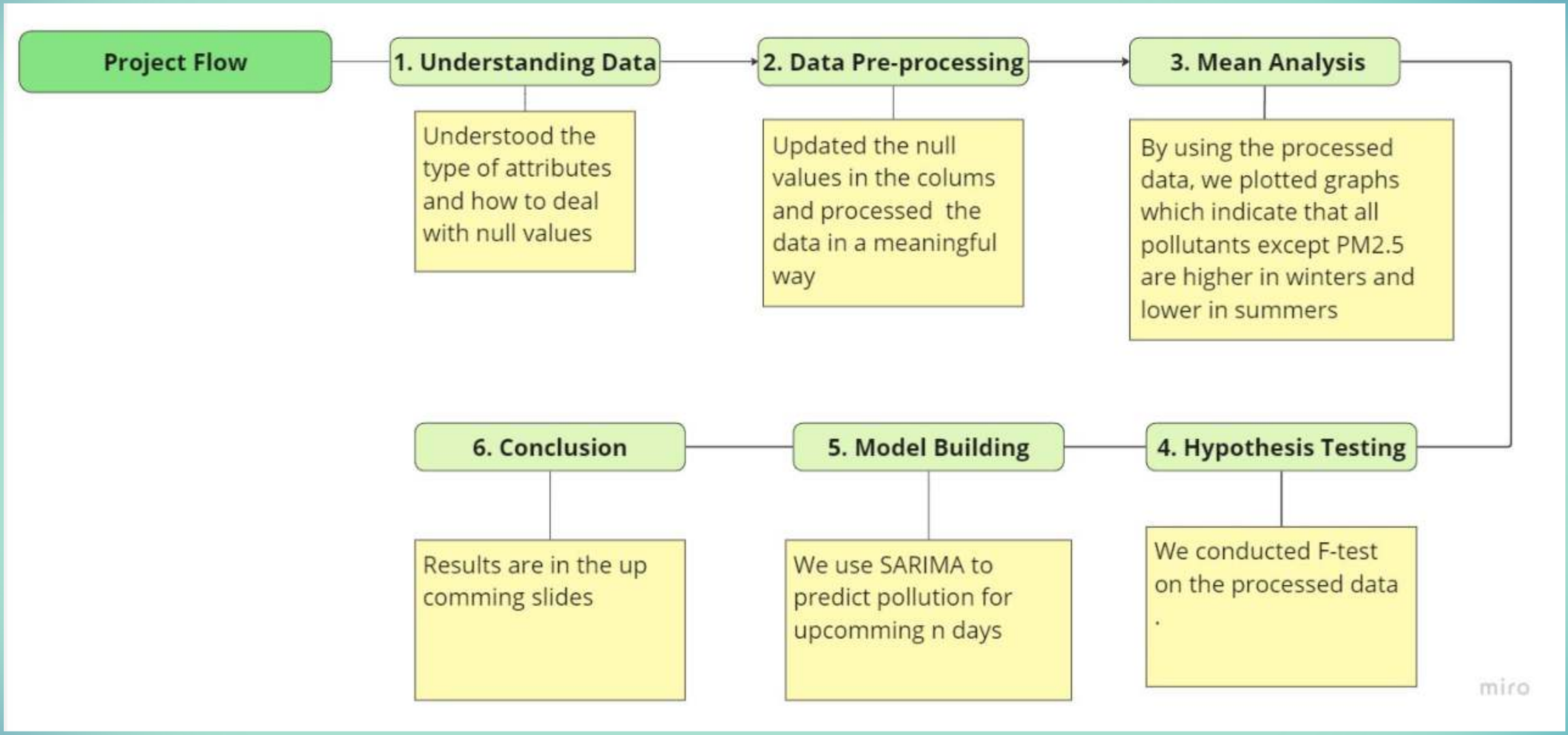
- The problem statement focuses on addressing data challenges, exploring correlations, and developing predictive models to better understand air quality in Beijing, with an emphasis on the impact of meteorological conditions, rain, temperature, pressure, and wind direction.

- This research is crucial for improving environmental policy, public health, and predictive models to mitigate air pollution's adverse effects on the city's residents.

Importance of the project

- **Public Health:** The project can help protect the health of Beijing's residents by identifying pollution trends and their impact on health outcomes, leading to informed health policies.
- **Environmental Policy:** It can inform the development and evaluation of environmental policies to improve air quality and reduce pollution levels in the city.
- **Predictive Models:** By developing predictive models, the project contributes to early warning systems, allowing the city to prepare for and respond to air quality issues proactively.

Project Flow



Hypothesis Tests

- **Null Hypothesis (H0)**
  - The model with four features i.e. temperature, pressure, dew point temperature, and rain is not significantly better than the model built with a proper subset of these features on influencing pollution levels (PM2.5).

- **Alternative Hypothesis (H1)**
  - The complex model is significantly better than the simple model on influencing pollution levels (PM2.5).

F-Test

- We built a linear regression model with 4 features - TEMP, PRES, DEWP, and RAIN and performed an F-Test to check if a model built by using a subset of these features is close to the performance of the complex model (containing all the features).

- Attached pics are the formulae of F-test used to analyse two models.

$$Residual\ Sum\ of\ Squares\ (RSS) = \sum_{i=0}^{n} (y_i - \hat{\mu}_i)^2$$

RSS_1 = Residual Sum of Squares of fitted model 1

RSS_2 = Residual Sum of Squares of fitted model 2

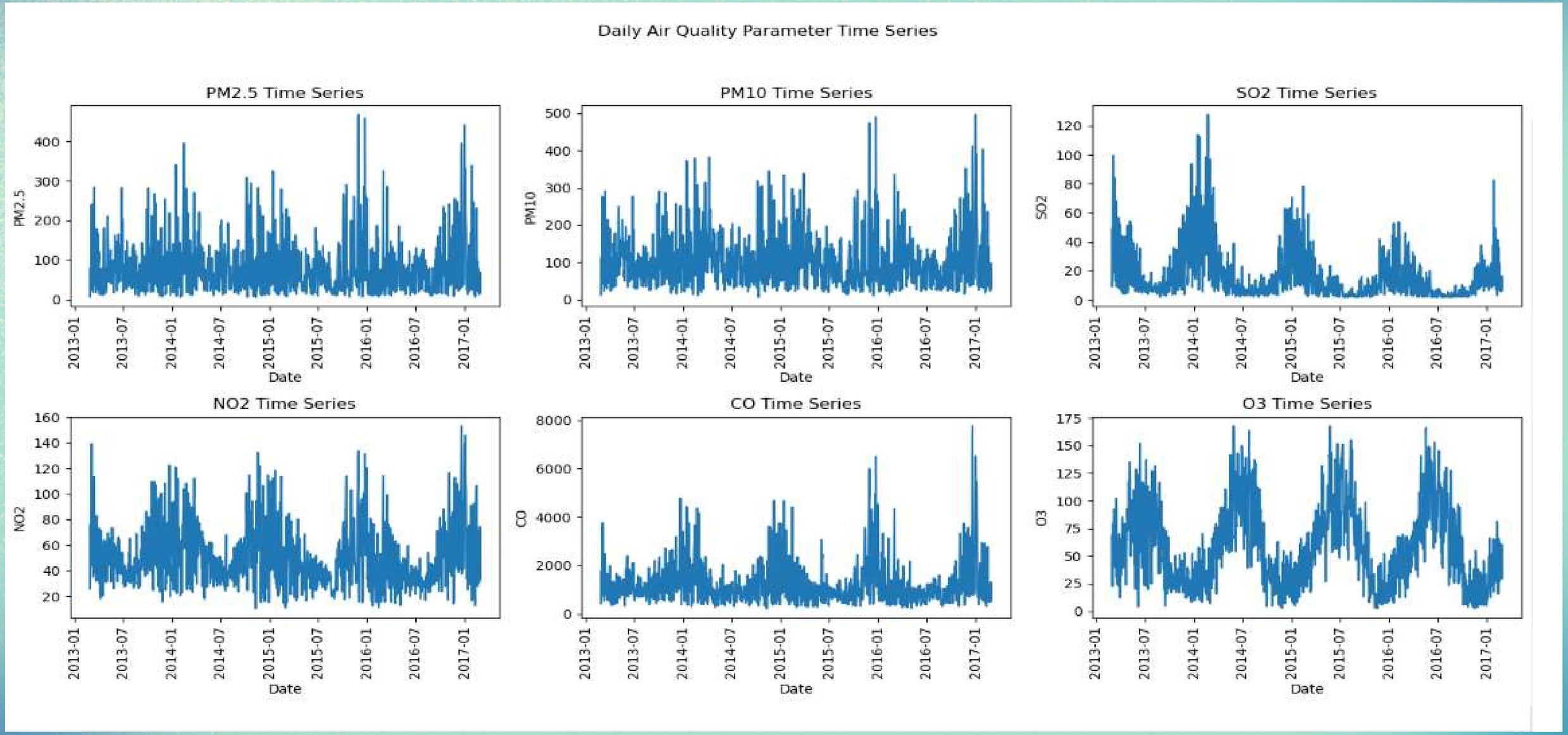
$$F\ statistic = \frac{(\frac{RSS_1 - RSS_2}{k_2 - k_1})}{(\frac{RSS_2}{n - k_2})}$$

Analysis

- After doing the F-test, we observed that the complex model is better than all the models(simpler models) built using the proper subset of the features used for building the complex model.

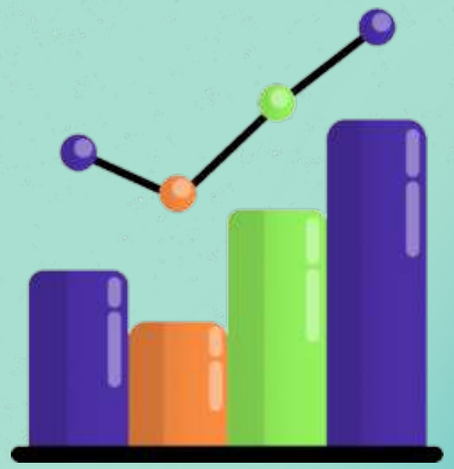
- Using the refined dataset, we generated graphical representations that clearly demonstrate elevated levels of all pollutants during the winter months and decreased levels during the summer season,except for the feature PM2.5.

Mean Analysis



Modeling

- **ARIMA (AutoRegressive Integrated Moving Average)** and **SARIMA (Seasonal AutoRegressive Integrated Moving Average)** are used to forecast the time series models.
- Firstly, we used ARIMA (AutoRegressive Integrated Moving Average) modeling, but the disadvantage of the ARIMA model is that it’s unable to capture complex seasonality.
- So, we used the SARIMA (Seasonal AutoRegressive Integrated Moving Average), which extends ARIMA by incorporating seasonal components into the model. It includes parameters for seasonal differences, seasonal autoregressive terms, and seasonal moving average terms.



```python
import pmdarima as pm

# Assuming 'train' contains your training data for pollutants and features
selected_features = ['TEMP', 'PRES', 'DEWP', 'RAIN', 'wd', 'WSPM']
pollutants = ['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3']

for pollutant in pollutants:
    # Train auto_arima separately for each pollutant
    smodel = pm.auto_arima(train[pollutant], exogenous=train[selected_features],
                           seasonal=True, m=12,
                           stepwise=True,  # Use stepwise search
                           trace=True)     # Print detailed results

    # Get the best parameters for the current pollutant
    print(f"Best parameters for {pollutant}:")
    print(smodel.summary())
```

This code captures the best values for the parameters that are used in the SARIMA model.

```
order=(2, 1, 0), seasonal_order=(2, 0, 0, 12)
```

Modeling Using SARIMA

```python
for pollutant in pollutants:

    model = sm.tsa.statespace.SARIMAX(train[pollutant], order=(2, 1, 0), seasonal_order=(2, 0, 0, 12))
    sarima_model = model.fit()

    # Forecast the next 10 days
    forecast = sarima_model.get_forecast(steps=10)
    predicted_values = forecast.predicted_mean

    # Print or store the predicted values
    print(f"Predicted values for {pollutant}:")
    print(predicted_values)
```

Python

- The **SARIMA** model uses the past and present-day data to predict the pollution of next N days

Original Values

```
test
```

	TEMP	PRES	DEWP	RAIN	wd	WSPM	PM2.5	PM10	SO2	NO2	CO	O3
420758	14.6	1013.3	-15.6	0.0	N	3.6	5.0	5.0	4.0	8.0	100.0	117.0
420759	15.4	1013.0	-15.0	0.0	NNW	3.3	6.0	19.0	4.0	8.0	100.0	122.0
420760	14.9	1012.6	-15.4	0.0	NW	2.1	12.0	18.0	5.0	9.0	200.0	122.0
420761	14.2	1012.5	-14.9	0.0	NW	3.1	12.0	23.0	6.0	13.0	200.0	120.0
420762	13.4	1013.0	-15.5	0.0	WNW	1.4	13.0	29.0	5.0	22.0	300.0	109.0
420763	12.5	1013.5	-16.2	0.0	NW	2.4	12.0	29.0	5.0	35.0	400.0	95.0
420764	11.6	1013.6	-15.1	0.0	WNW	0.9	13.0	37.0	7.0	45.0	500.0	81.0
420765	10.8	1014.2	-13.3	0.0	NW	1.1	16.0	37.0	10.0	66.0	700.0	58.0
420766	10.5	1014.4	-12.9	0.0	NNW	1.2	21.0	44.0	12.0	87.0	700.0	35.0
420767	8.6	1014.1	-15.9	0.0	NNE	1.3	19.0	31.0	10.0	79.0	600.0	42.0

Predictions

```
Predicted values for PM2.5:
10000     8.907935
10001     8.638265
10002     8.626207
10003     9.402948
10004     9.799539
10005    10.003196
10006    10.514060
10007    10.827994
10008    11.458262
10009    11.384214
Name: predicted_mean, dtype: float64
```

```
Predicted values for PM10:
10000    22.856529
10001    22.570066
10002    23.186334
10003    24.226382
10004    24.201773
10005    24.890225
10006    24.011776
10007    25.830200
10008    26.348834
10009    25.792790
Name: predicted_mean, dtype: float64
```

```
Predicted values for SO2:
10000    7.176054
10001    7.177522
10002    7.276499
10003    7.372537
10004    7.496450
10005    7.620253
10006    7.536648
10007    7.384983
10008    7.329244
10009    7.289056
```

```
Predicted values for NO2:
10000    20.954621
10001    20.853359
10002    21.003725
10003    22.463454
10004    22.715054
10005    23.938182
10006    24.390667
10007    25.212013
10008    25.567656
10009    28.755004
```

```
Predicted values for CO:
10000    199.593865
10001    199.680038
10002    200.578527
10003    210.488742
10004    220.745387
10005    230.259212
10006    230.682229
10007    240.193450
10008    240.607700
10009    268.312261
```

```
Predicted values for O3:
10000    113.109185
10001    116.830686
10002    119.169365
10003    113.987394
10004    112.136127
10005    106.931646
10006    103.135181
10007     97.444181
10008     95.002426
10009     86.083433
```

Conclusion

- Preprocessed the data, imputed new features, and eliminated irrelevant data.
- Used the F test to see if all the features are relevant or not.
- Used time series data to forecast the pollution level.
- Trained SRIMA to forecast the pollution for the upcoming n days.